Remarks

In the Office Action dated February 16, 2007 ("Office Action"), Claims 1-22 were rejected. In the amendment set forth above, Claim 9 is amended, and the remaining Claims are unchanged. In view of the amendments to the Claims and the comments set forth below, it is respectfully submitted that all Claims are in condition for allowance.

1. Claims 9-16 were rejected under 35 USC §101 as being directed to non-statutory subject matter. Independent Claim 9 has been amended to recite that the instructions stored on the digital medium are computer-readable, as suggested by the Examiner. With this change, it is submitted that Claim 9, and dependent Claims 10 – 16 that depend from Claim 9, satisfy the requirements of 35 USC §101, and it is requested that this rejection be withdrawn.

2. Claims 1-4, 6, 9-12, 14, and 17-21 were rejected under 35 USC §103(a) as being unpatentable over U.S. Pat. App. Pub. No. US2001/0001157 to Oura ("Oura") in view of U.S. Pat. No. 6,446,241 to Mobley et al. ("Mobley"). This rejection is respectfully traversed.

Applicants' Claim 1 describes a simulation mechanism that uses software (referred to as Function Generating Programs, or FGPs) to prepare a test source file for simulating a cache memory. (Claim 1 line 2.) This test source file will be executed during simulation to verify a circuit design before any actual device is built using that design. Applicants' Specification makes clear that this simulation process <u>does not</u> in any way related to testing <u>actual integrated circuit devices</u>, as follows:

> "It should be recognized that <u>the FGP does not actually write to or update a cache memory location. It merely creates functions that are used by the simulator to write to the cache memory location maintained by the simulator</u>. The data integrity buffer is likewise not a part of the simulator, but a separate buffer created by the FGP whenever it runs, and closed by

the FGP when the FGP stops running and the FGP's output file, the test source file, is complete. (Specification p. 25 paragraph 122.)"

In accordance with the description of the invention in the Specification, the Claim 1 preamble makes clear that Claim 1 relates to verifying the performance of a simulated cache memory device. (Claim 1 line 2.) Moreover, Claim 1 line 4 describes the step of "sequentially and randomly creating a series of functions" wherein a function is defined in Applicants' Specification as follows:

> A "function" is defined as a command given to a hardware or software simulator to direct the simulator to perform a predetermined activity. (Specification page 7 paragraph 37.)

The Examiner cites Oura as teaching Applicants' step of randomly creating a series of functions (i.e., simulator commands). Applicants respectfully disagree. In contrast to Applicants' step of creating a series of functions (simulator commands), Oura does not relate in any way to creating any series of functions for simulation. Oura describes a system for testing a physical device. That is, in Oura, an actual manufactured device shown as apparatus 31 (Oura Figure 1) is coupled to the cache testing device 10. The cache testing device 10 then tests the embedded cache of apparatus 31 to determine whether manufacturing defects exist. (See Oura Figure 1.) In particular, the Oura cache testing device 10 provides a way to test very large cache memories using a pseudo-random mechanism that does not duplicate an access to a same cache location that has already been tested. This allows testing of the cache to be completed more quickly. This is discussed in Oura as follows:

> "According to the above invention, individual blocks of the cache can be verified (initialized) without duplication. Hence, by using this cache testing apparatus, the cache can be verified at high speed and securely." (Oura paragraph 15, emphasis added.)

Oura describes that an actual device is being tested, as follows:

> "Referring now to FIG. 4 and FIG. 5, in an example of an apparatus to be

> tested being an information processing apparatus having 256 entries and two-way cache, the operation of the cache testing apparatus at the time of command row creating process for initializing the cache is explained more specifically below." (Oura paragraph 64, emphasis added.)

Thus, unlike Applicants' system of Claim 1, Oura is not related in any way to simulating a design to locate logical design faults, but rather is a device for testing for flaws in an actual manufactured device. This is why Oura is categorized in class/subclass 714/42, which relates to detecting a component fault in an actual memory storage device.

The objectives, techniques, and methodologies used to simulate a logic design are much different from those associated with testing an actual device. When simulating a design, tests are made to determine whether unexpected conditions will occur. For instance, Applicants' simulation program tests that requests that should not change the state of the cache do not, in fact, change that state because of some unintended logical error in the design This is why Applicants' invention intersperses non-access commands with cache access commands in a random fashion in Applicants' program, as described in the creating step of Applicants' Claim 1. (See also Specification page 3 paragraph 11.) In contrast, Oura only needs to test access commands to various addresses to determine whether a manufacturing defect exists in the memory cell. This is why in Oura, all commands that are not access commands are ignored for testing purposes. (See Oura Figure 12, and in particular the "no" path of step 403.)

For the foregoing reasons, Oura is completely unrelated to a simulation system as described in Applicants' Claim 1 preamble and Applicants' creating step. Oura does not teach or even suggest Applicants' step of sequentially and randomly creating a series of functions, which are commands to be simulated. For at least this reason, this rejection is improper.

Turning next to the remaining steps of Claim 1, these steps include:

> updating a data integrity buffer after each function of said series of functions is created;
> creating a series of integrity check functions from said data integrity buffer; and
> writing said series of functions and said series of integrity check functions to a test file.

As clearly described in Applicants' Specification, the integrity check functions provide the results that are expected (assuming proper design operation) after the series of functions are simulated. That is, the integrity check functions provide the expected contents of the cache after each function in the test file is completed. This is described in Applicants' Specification as follows:

> "Having written the function to the output buffer, the FGP then updates the data integrity buffer in block **556**. The data integrity buffer can be thought of as a collection of cache memory addresses associated with the data they hold. To build the data integrity buffer, the FGP examines the function it has just assembled and written to the output buffer. It identifies the memory addresses of the function and the data that are stored there. It makes an entry in the buffer for this memory location, recording both the location and the data stored there. ...The FGP shadows the cache memory write operations of the FGP, <u>maintaining in the data integrity buffer the very latest contents of each of cache memory locations.</u> Whenever a new function writes to a location, the FGP either adds that new location to the data integrity buffer, or (if the location already exists in the data integrity buffer) it replaces the existing contents of that location with the latest contents written to that location." (Specification page 24 paragraph 118 and page 25 line 119, emphasis added.)

During simulation of a design, the series of functions that have been stored to the test file are first retrieved from that file and issued as commands to the simulator that is simulating the design. Then the integrity check functions are likewise retrieved from that same file and used to determine whether the design simulation completed as expected. In other words, the simulated state of the

cache is compared to the cache state reflected by the test file to determine whether a fault was detected in the design. This is described in the Specification, as follows:

> "The integrity check functions cause the simulator to read the data from the cache memory locations and to compare that data to data in the integrity check functions. The data in the integrity check functions are the data earlier stored in the data integrity buffer when the FGP was creating the test file now being executed by the simulator. After comparing the contents of the cache memory location with the data in the integrity test functions, the simulator is configured to indicate whether the two are or are not equal – in other words, whether the cache memory location data is correct or not. If it is not correct, the cache is deemed to have failed the test ." (Specification p. 28 paragraphs 138 and 139.)

The Examiner states that the updating, creating, and writing steps of Claim 1 lines 5-10, which describe the integrity check functions discussed above, are taught by Mobley. Applicants' disagree. The Mobley system does not store expected simulation results in any file prior to the start of simulation. Instead, Mobley issues simulation tests to two design models at once. These two design models include the cache/memory system design model that is being tested, and a second "flat memory reference" design that is assumed to be correct. Both models are simulated at the same time, and it is determined whether the two models produce the same simulation result, as follows:

> "Generally the test will apply the same stimulus to cache/memory system 210 as to flat memory reference model 220 to see if they produce the same results. Flat memory reference model 220 is assumed to be correct since it is so simple. Therefore differing results indicates a design error in cache/memory system. (Mobley col. 7 lines 3-8.)

The details of the Mobley system are described as follows:

> "Data stream stimulator 241 contains a dual-port sparse memory which is loaded with the data thread of the test. Data stream stimulator 241 reads two commands out every cycle and applies one or both commands to the two load-store interfaces of the level one data cache. Data stream

stimulator 241 also applies these commands to flat memory reference model 220....The monitors 235, 245 and 255 merely watch for data being returned from cache/memory system 210 and compare that to the data being returned from flat memory reference model 220. If they differ, an error is flagged so that the cache system designers can debug it." (Mobley column 8 lines 19-24 and 56-60, emphasis added.)

Thus, in Mobley, the expected simulation results are not obtained from any file that was prepared in advance of simulation. Rather, the expected simulation results are generated by simulating a second design model that is known to be correct, and that is being simulated at the same time as the model-under-test. In fact, Mobley very clearly states that the files containing the Mobley test transactions (that is, the "transaction files") don't contain any expected test results as follows:

"These transaction files don't contain the expected data, nor do the transaction files specify the exact cycle in which the stimulus is applied." (Mobley col. 8 lines 2-4, emphasis added.)

Thus, Mobley clearly states that in Mobley, the test files don't contain integrity check functions, as is claimed by Applicants' updating, creating, and writing steps of Claim 1. Instead, the integrity data is obtained from a second simulation being conducted at the same time as the simulation of the design-under-test.

The Mobley system requires special considerations. For instance, a special dual port is needed to perform the parallel simulation, as described by Mobley. (See the above Mobley passage). In addition, Mobley requires that a second flat design model that is known to be correct be available to generate the expected results. Applicants' method does not require these additional elements, and is therefore arguably less complex. For the reasons set forth above, Mobley does not teach the steps of Applicants' Claim 1 related to integrity check functions, and this rejection is improper for this additional reason.

Before continuing, and for completeness sake, the Examiner's specific citations appearing on page 4 of the Office Action regarding the teachings of Mobley are next considered. The Examiner states that Mobley column 3 lines 1-

6 teach the updating and creating steps of lines 4-8, and Mobley column 13 lines 40-50 and column 14 lines 15-20 teach the writing step. These assertions are considered in turn.

Mobley column 3 lines 1-6 appear as follows:

> "The series of test sequences are applied to the control logic cache design and to a reference memory. The results are compared. If the response of the control logic cache design fails to match the response of the reference memory then a design fault is detected."

The cited Mobley passage reiterates that in Mobley, the design-under-test and a second reference memory model are simulated at the same time, and the results of the two different simulations are compared to determine whether a failure occurred. In Mobley, the expected results are not stored in a file prior to simulation, as is described by Applicants' Claim 1.

Next, Mobley column 13 lines 40-50 are considered. This passage is as follows:

> "Test generation for the cache can be done at several levels. In the preferred embodiment a low level test language is created for the cache test bench. A low level test language assembler processes source files written with this language into a set of synopsis memory image files suitable for execution on cache system test bench 200. The low level test language source files can be written by hand or generated by any kind of automatic test generator. In particular, a random test pattern generator called mguber is preferably used to generate cache tests."

This passage relates to how the simulation tests are generated, and does not relate to how any expected results of those tests are generated or used. Thus this passage, like the previous passage, does not teach Applicants' steps related to integrity checks.

Finally, Mobley column 14 lines 15-20 appear as follows:

> "The results of the cache tests are formed in a cache tree file. This file gives a line-by-line list of transactions within the cache, giving only vital information about each transaction. It is intended for initial debugging before the location or nature of a failure is known."

This passage describes how the actual results of the simulation are stored for analysis. This does not relate to storing of any expected results (e.g., the integrity check functions). As previously stated, in Mobley, the expected results are not generated or stored to a file in the manner recited in Claim 1. Instead, the expected results are generated during a parallel simulation that occurs at the same time as the design-under-test is being simulation. It may also be noted from the cited passage that the "cache tree file" in which the actual results are stored is not the same file used to store the original tests. For this additional reason, this passage does not teach Applicants' Claim 1, which recites that the series of functions and the series of integrity check functions are both stored to the test file.

To summarize, nothing in Mobley describes Applicants' method of updating a data integrity buffer after each function is created, creating a series of integrity check functions from that buffer, and then writing those integrity check functions to the test file along with the functions. Moreover, Oura adds nothing to Mobley to teach these steps. For this additional reason, the rejection of Claim 1 based on Oura in view of Mobley is improper, and should be withdrawn.

Finally, the rejection of Claim 1 is improper because one skilled in the art would not be motivated to combine aspects of Mobley with Oura. As discussed above, Oura relates to a system for testing actual devices classified under class/subclass 714/42 relating to detection of component faults. In contrast, Mobley is classified under 716/4, which involves design analysis testing of semi-conductor masks (e.g., simulation). As previously discussed, the goals and techniques of testing a design for logical errors as taught by Mobley are entirely different from the goals and techniques of testing an actual manufactured device for manufacturing defects, as described in Oura. One skilled in the art would

have absolutely no motivation to combine aspects of the Mobley simulation system with the Oura device tester. In fact, such a combination would in all likelihood be inoperable. For example, the goal of the Oura system is to very rapidly test large caches embedded within actual devices. Attempting to do this by comparing actual cache contents of an Oura device with the results of a working simulation model as taught in Mobley would defeat the goal of the Oura system for at least the reason that executing the Mobley simulation model would be far slower than performing accesses to the actual cache device as being done in Oura. Moreover, personnel involved in testing of manufactured devices (e.g., testing personnel at a manufacturing facility) almost certainly do not have access to a simulation reference model of the type required by Mobley. The type of a Mobley simulation model is generally only used during design phases, and not for testing actual parts.

In conclusion, the rejection of Claim 1 based on Oura in view of Mobley is improper for at least the following reasons:

a.) Nothing in Oura teaches or suggests Applicants' step of sequentially and randomly creating a series of functions (which are defined as commands given to a simulator) for at least the reason that Oura is not related in any way to simulation.

b.) Nothing in Mobley teaches or suggests Applicants' updating, creating, and writing steps of Claim 1 lines 5-10. In fact, Mobley teaches a very different method of verification involving use of a second design model. Nothing in Oura adds anything to Mobley to teach or suggest this step.

c.) One skilled in the art would not be motivated to make the cited combination of references, and the resulting system would likely be inoperable. At the very least, the goals of Oura would be defeated by incorporation of aspects of the Mobley system with Oura.

For at least the foregoing reasons, the rejection of independent Claim 1 is improper, and should be withdrawn.

Independent Claims 9 and 17 include aspects of the invention that are

similar to those set forth in regard to Claim 1. These Claims are allowable over this rejection for reasons similar to those discussed in regards to Claim 1.

Dependent Claims 2-4 and 6 depend from Claim 1, dependent Claims 10-12 depend from Claim 9, and dependent Claims 18-21 depend from Claim 17. These dependent Claims are allowable over the current rejection for at least the reasons set forth in regards to Claim 1. These Claims include additional aspects not taught or suggested by the cited combination of references, and these Claims are allowable over the rejection for additional reasons related to these aspects.

3. Claims 5 and 13 were rejected under 35 USC §103(a) as being unpatentable over Oura in view of Mobley, and further in view of U.S. Pat. No. 5,386,579 to Bourekas et al. ("Bourekas"). This rejection is respectfully traversed.

Dependent Claim 5 depends indirectly from Claim 1, and dependent Claim 13 depends indirectly from Claim 9. These dependent Claims are allowable over the current rejection for at least the reasons set forth above in regards to Claim 1. Nothing in Bourekas adds anything to the other references to teach or even suggest the various aspects of Applicants' independent Claims 1 and 9. Moreover, Bourekas relates to a multiplexed address/data bus system. One skilled in the art would not be motivated to combine aspects of Bourekas with the Oura device testing system and/or the simulation system of Mobley. For this additional reason, the rejection is improper, and should be withdrawn.

4. Claims 7, 15, and 22 were rejected under 35 USC §103(a) as being unpatentable over Oura in view of Mobley, and further in view of U.S. Pat. Pub. No. US 2004/0181655 to Azuma ("Azuma"). This rejection is respectfully traversed.

Dependent Claim 7 depends indirectly from Claim 1, dependent Claim 15 depends indirectly from Claim 9, and dependent Claim 22 depends indirectly from Claim 17. These dependent Claims are allowable over the current rejection

for at least the reasons set forth above in regards to Claim 1. Nothing in Azuma adds anything to the other references to teach or even suggest the various aspects of Applicants' independent Claims. Moreover, Azuma relates to a signal processor. One skilled in the art would not be motivated to combine aspects of Azuma with the Oura device testing system and/or the simulation system of Mobley. For this additional reason, the rejection is improper, and should be withdrawn.

5. Claims 8 and 16 were rejected under 35 USC §103(a) as being unpatentable over Oura in view of Mobley, and further in view of publication entitled "The Art of Computer Programming" by Donald E. Knuth, vol. 2 ("Knuth").. This rejection is respectfully traversed.

Dependent Claim 8 depends indirectly from Claim 1, and dependent Claim 16 depends indirectly from Claim 9. These dependent Claims are allowable over the current rejection for at least the reasons set forth above in regards to Claim 1. Nothing in Knuth adds anything to the other references to teach or even suggest the various aspects of Applicants' independent Claims. For this additional reason, the rejection is improper, and should be withdrawn.

Conclusion

In the Office Action dated February 16, 2007, Claims 1-22 were rejected. In the amendment set forth above, Claim 9 was amended and the remaining Claims are unchanged. In view of the amendments to the Claims and the comments set forth below, it is respectfully submitted that all Claims are in condition for allowance. Therefore, a Notice of Allowance is respectfully requested. If the Examiner has questions or comments regarding any of the foregoing, a call to the undersigned is encouraged and welcomed.

Respectfully submitted,

Beth L. McMahon 6/11/2007

Beth L. McMahon
Attorney for Applicants
Reg. No. 41,987
Tele No. (651) 635-7893

Unisys Corporation
M.S. 4773
P.O. Box 64942
St. Paul, MN 55164-0942